A Critical Evaluation of Several Global Optimization Algorithms for the Purpose of Molecular Docking

DAVID J. DILLER,* CHRISTOPHE L. M. J. VERLINDE

Department of Biological Structure, Biomolecular Structure Center, University of Washington, Box 357742, Seattle, Washington 98195

Received 21 April 1999; accepted 17 August 1999

ABSTRACT: Several global optimization algorithms were applied to the problem of molecular docking: random walk and Metropolis Monte Carlo Simulated Annealing as references, and Stochastic Approximation with Smoothing (SAS), and Terminal Repeller Unconstrained Subenergy Tunneling (TRUST) as new methodologies. Of particular interest is whether any of these algorithms could be used to dock a database of typical small molecules in a reasonable amount of time. To address this question, each algorithm was used to dock four small molecules presenting a wide range of sizes, degrees of flexibility, and types of interactions. Of the algorithms tested, only stochastic approximation with smoothing appeared to be sufficiently fast and reliable to be useful for database searches. This algorithm can reliably dock relatively small and fairly rigid molecules in a few seconds, and larger and more flexible molecules in a few minutes. The remaining algorithms tested were able to reliably dock the small and fairly rigid molecules, but showed little or no reliability when docking large or flexible molecules. In addition, to decrease the error in the typical grid-based energy evaluations a new form of interpolation, logarithmic interpolation, is proposed. This interpolation scheme is shown to both quantitatively reduce the numerical error and practically to improve the docking results.
© 1999 John Wiley & Sons, Inc. J Comput Chem 20: 1740–1751, 1999

Keywords: global optimization; docking; stochastic approximation with smoothing; terminal repeller unconstrained subenergy tunneling; logarithmic interpolation

*Current address: Pharmacopeia, Inc., CN5350, Princeton, NJ 08543-5350

Correspondence to: C. L. M. J. Verlinde; e-mail: verlinde@gouda.bmsc.washington.edu

CCC 0192-8651 / 99 / 161740-12

Introduction

The docking problem falls into the general and large category of global optimization problems, that is, problems that involve finding the global minimum or maximum of a target function. As virtually every scientific and engineering discipline has many important problems that fall into this category, there is a wide range of global optimization algorithms. Most optimization algorithms fall roughly into one of three classes: gradient-based algorithms (steepest descent, conjugate gradient, Newton's method, etc.), stochastic algorithms (Metropolis Monte Carlo simulated annealing, genetic algorithms, etc.), and combinatorial algorithms.

The strength of the gradient-based methods is that given a good starting point, they find the global optimum rapidly. Unfortunately, the docking problem is one in which a good starting point is generally not known. As a result, these algorithms often get trapped in a local minimum. Thus, by themselves, the gradient-based methods do not offer a useful solution to the docking problem.

The stochastic algorithms have the advantage that they offer the chance that any given choice of the parameters will be sampled. Many of these algorithms are guaranteed, if given enough time, to get arbitrarily close to the global optimum. These algorithms have the disadvantage that they often require a large amount of CPU time to achieve an acceptable degree of reliability. The stochastic algorithms that have been most successfully applied to the docking problem are either physically motivated, Metropolis Monte Carlo simulated annealing (AutoDock 2.4 and earlier versions,[1] BoxSearch,[2] etc.), or biologically motivated, genetic algorithms (GOLD,[3] AutoDock 3.0[4]). These algorithms, however, only scratch the surface of the global optimization algorithms in this class.

Combinatorial algorithms have the advantage that they can be extremely fast and effective. Unfortunately, for problems with continuous variables, approximations to the search space are necessary. Likely, the most successful combinatorial algorithms used for molecular docking are distance geometry/matching (DOCK[5,6] and FLOG[7]) and incremental construction (FlexX[8]). These algorithms have particularly set themselves apart in their ability to dock a database of small molecules to a target protein in a reasonable amount of time.

In this article we address the question: can the continuous methods (stochastic and/or gradient based) be used to reliably dock a database of typical small molecules to a target protein in a reasonable amount of time. Reliable docking depends on both an adequate scoring function, whose global minimum corresponds to the biologically relevant complex, and an optimization algorithm that consistently finds that global minimum. Here we focus on the optimization aspect. By reliable we mean that for any given small molecule there should be a 99% chance that the molecule has been docked to within 5% of its global minimum. The choice of 5% is somewhat arbitrary, but conservatively reflects the error of most scoring functions. By a typical small molecule, we mean any molecule that would be considered a decent lead compound (assuming sufficient activity) in a drug design project. In particular, this excludes anything too large (molecular weight ≥500) or too flexible (number of rotatable bonds ≥8). Finally, by a reasonable amount of time we mean that an algorithm should be able to dock a database of around 10,000 small molecules in a week.

To address this question, we compare the results of docking a variety of compounds using four different algorithms: random walk. Metropolis Monte Carlo simulated annealing,[9] stochastic approximation with smoothing,[10,11] and terminal repeller unconstrained subenergy tunneling.[12,13]

In particular, we are interested in the time required to gain various levels of reliability with each algorithm and the dependence of these times on the size and flexibility of the small molecule being docked.

Methods

THE SCORING FUNCTION

The scoring function used in this study is

$$S = \sum_{\text{Protein}} \sum_{\text{Ligand}} \left(\frac{R_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}} + \frac{q_i q_j}{\varepsilon(r_{ij}) r_{ij}} \right) + \sum_{\text{HBonds}} \left(\frac{B_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{10}} \right) + \sum_{\text{Ligand}} \sum_{\text{Ligand}} \left(\frac{R_{ij}}{r_{ij}^{12}} - \frac{A_{ij}}{r_{ij}^{6}} \right) + \sum_{\text{Dihedrals}} \gamma_k \left(1 + \cos\left(\omega_k \theta_k - \theta_k^0\right)\right). \tag{1}$$

It consists of standard 12-6 van der Waals potentials, electrostatics modeled via Coulomb's law with a sigmoidal distance dependent dielectric constant,[14] $\varepsilon(r)$, a 12-10 potential to model hydrogen bonds, and a dihedral term. This scoring function

is essentially that used by AutoDock 2.4[1] supplemented with a dihedral term, i.e., the final term on the right-hand side of eq. (1). The reasons for choosing this scoring function are that it has been extensively tested, and this type of scoring function is used in many docking studies. In the Relevance and Distribution section we show that this scoring function is sufficiently accurate for our purposes.

The second aspect of the AutoDock scoring scheme employed in this study is that the intermolecular interactions are precalculated on grids (one atomic affinity grid for each atom type and one electrostatic grid). During the docking simulation the intermolecular score is approximated from the grids. This approach has the advantage that the score is extremely efficient to calculate. The disadvantage of the grid-based approach is that if an atom is not exactly on a grid point a tri-linear interpolation between the eight surrounding grid points is performed. Due to the steepness of the van der Waals potentials the linear interpolation often results in considerable error. To decrease this error, a logarithmic interpolation was used in this study rather than a linear one. The difference between these two interpolation schemes for a function of one variable is summarized as follows. For $0 < x < 1$, the linear interpolation approximates $f(x)$ via

$$f(x) \approx xf(1) + (1 - x)f(0). \tag{2}$$

In contrast, the logarithmic interpolation approximates $f(x)$ via

$$f(x) \approx -RT \ln\left(xe^{-f(1)/RT} + (1 - x)e^{-f(0)/RT}\right) \tag{3}$$

where, in general, RT is any positive constant. The constant RT is used to suggest interpolation in probability space rather than in energy space, although for this work a value of $RT = 2.0$ was used. In general, the logarithmic interpolation will fall between $f(0)$ and $f(1)$, but will be weighted closer to the minimum of $f(0)$ and $f(1)$ than the linear interpolation. Thus, the logarithmic interpolation is always less than the linear interpolation. Indeed, as $RT \to 0$, the logarithmic interpolation approaches $\min\{f(1), f(0)\}$, and as $RT \to \infty$, the logarithmic interpolation approaches the linear interpolation. The difference between these two types of interpolation for a function of one variable is shown in Figure 1a for the standard 12-6 van der Waals potential between two carbon atoms.

The major disadvantage of the linear interpolation is that it pathologically overestimates the van der Waals radii (Fig. 1). Logarithmic interpolation is more accurate in regions where the function is convex but worse in regions where the function is concave. Thus, the logarithmic interpolation was used only for the atomic affinity grids not for the electrostatic grid. In the case of the atomic affinity grids, the logarithmic interpolation is more accurate but can still result in considerable error (Fig. 1b). Logarithmic interpolation tends to underestimate the score, whereas the linear interpolation tends to overestimate the score. Thus, the logarithmic interpolation makes the local minima larger, whereas the linear interpolation makes them smaller.

For this study all charges on the small molecules were added with an in-house program that uses the charge equilibration algorithm developed by Rappé and Goddard.[15] The charges on all the proteins were added using the prepare script ofAutoDock. All grid files were created using AutoGrid 2.0.[1] In all cases, a 0.5-Å grid spacing was used.

THE GLOBAL OPTIMIZATION ALGORITHMS

The first algorithm used is a random walk (RW) algorithm. This algorithm is intended as a control, i.e., if it works then the problem/tests are too easy. This algorithm generates a sequence of points, $X_1, X_2, \ldots, X_n, \ldots$, in parameter space in the following manner. Given the current position, X_n, the next point X_{n+1} is generated via a random perturbation of X_n. This process is continued for a fixed number of steps.

The second algorithm is Metropolis Monte Carlo simulated annealing[9] (MMCSA). The MMCSA algorithm generates a sequence of points, $X_1, X_2, \ldots, X_n, \ldots$, in parameter space in the following manner. Given the current point, X_n, a new point, Y_{n+1}, is generated via a random perturbation of X_n. If the score is an improvement ($S(Y_{n+1}) < S(X_n)$), the step is accepted ($X_{n+1} = Y_{n+1}$). If the score is not an improvement, a uniformly distributed random number, p, between 0 and 1 is generated. If

$$p < e^{-(S(Y_{n+1}) - S(X_n))/RT} \tag{4}$$

the step is accepted ($X_{n+1} = Y_{n+1}$), otherwise, the step is rejected ($X_{n+1} = X_n$). This process is continued for a fixed number of steps. Here, RT is a positive constant that is lowered periodically during each docking run to simulate an annealing process. This algorithm is essentially that employed by AutoDock 2.4.[1]

The third algorithm tested in this work is stochastic approximation with smoothing[10, 11] (SAS). This algorithm was initially used in circuit design. Heuristically, the target function is averaged to the extent that there is a single local minimum (Fig. 2a).

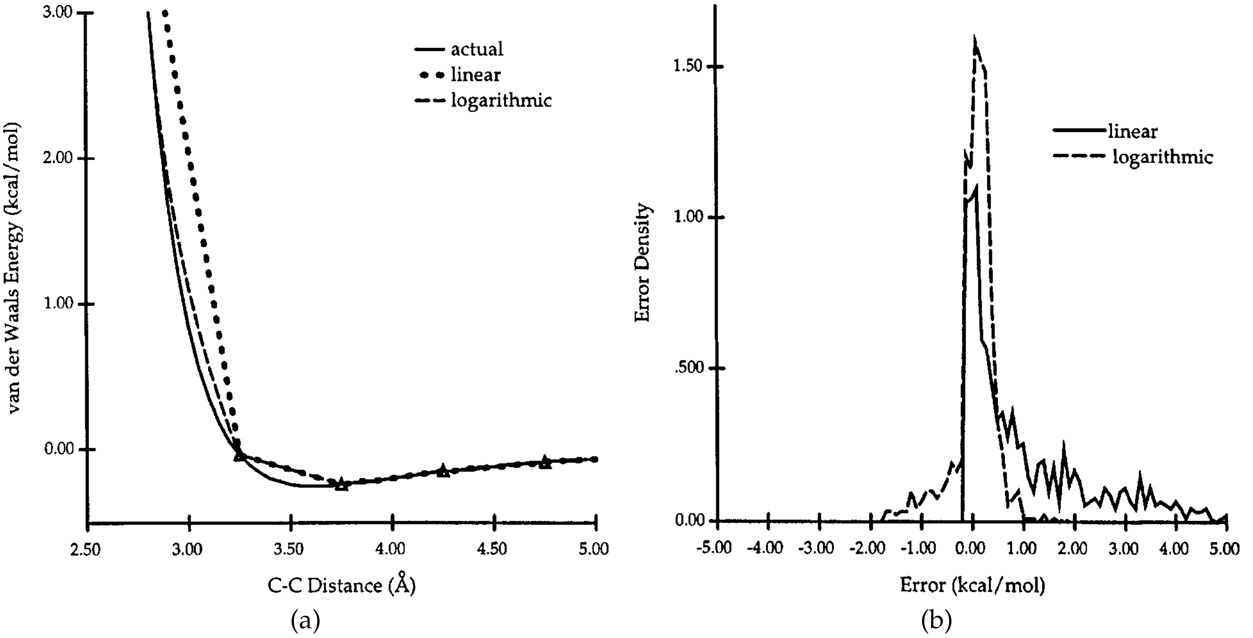

FIGURE 1. A comparison of linear interpolation to logarithmic interpolation. (a) Shown are the linear and logarithmic interpolation for the standard carbon–carbon van der Waals potential. The actual potential is the solid line, the linear interpolation is the dotted line, and the logarithmic interpolation is the dashed line. The triangles show the point from which the interpolation was done. A spacing of 0.5 Å was used. (b) Shown are the error density functions when using linear interpolation and logarithmic interpolation. The error distribution for the linear interpolation method is the solid line. The error distribution for the logarithmic interpolation is the dashed line. The curves should be interpreted as the probability that the error (x-axis) is between $E - \Delta E/2$ and $E + \Delta E/2$ is approximately $f(E) \times \Delta E$ where $f(E)$ is the value of the curve (y-axis) at the point E and ΔE is a small positive number. To calculate these distributions a $10 \times 10 \times 10$ Å^3 box around benzamidine in the benzamidine/β-trypsin structure was used with a 0.5 Å grid spacing. The value of the van der Waals interaction with a carbon atom was then computed at the center of each of the grid cubes in three ways: exactly, via linear interpolation from the eight surrounding grid points, and via logarithmic interpolation from the eight surrounding grid points. If the exact value of the carbon van der Waals interaction was above 2 kcal/mol, the point was ignored. The error at the remaining center points was included in each distribution. The main difference between the error distributions lies in the fact that the error for the linear interpolation is entirely to the right of 0, whereas the error arising from logarithmic interpolation is better balanced around 0. Thus, the linear interpolation scheme always overestimates the correct value, whereas the logarithmic interpolation gives a less biased approximation.

This minimum is then found via conjugate gradient minimization. Once this first minimum is found, the target function is averaged to a lesser degree (Fig. 2b). Again, conjugate gradient minimization is employed to optimize this function starting from the minimum found from the previously averaged function. This process is continued, each time averaging less than the previous. The hope is that by starting with a simple problem and gradually working to the more difficult problem at hand the global minimum will be easier to find (Fig. 2c). This algorithm is related to the Diffusion Equation Method[16,17] (DEM) used in the contexts of conformational analysis and protein folding, although there are significant implementation differences between the SAS and DEM algorithms. These differences are described below.

In mathematical terms, the SAS algorithm averages the target function as follows. For $\sigma > 0$, S_σ is defined as the convolution of S with a Gaussian smoothing function with standard deviation σ, that is,

$$S_\sigma(X) = \int S(Y) h_\sigma(X - Y)\, dY \tag{5}$$

where h_σ is the Gaussian smoothing function, i.e.,

$$h_\sigma(X) = (2\pi\sigma^2)^{-N/2} e^{-|X|^2/2\sigma^2}. \tag{6}$$

S is the target function, X is the multidimensional parameter, and N is the number of degrees of freedom. The integral in (5) is over all parameter space, which includes translations, rotations, and dihedral angles. The function given in (6) is certainly not the only function that could be used in eq. (5). The prop-

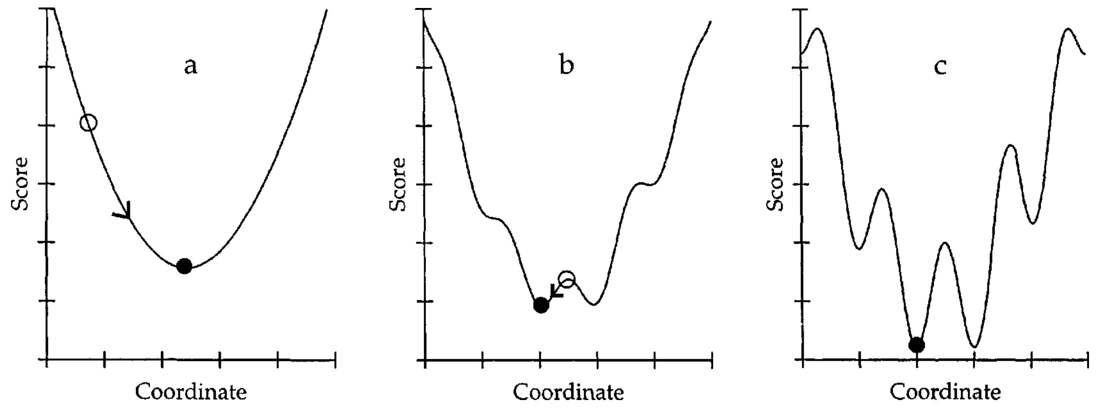

FIGURE 2. A schematic description of the SAS global optimization algorithm. (a) Shown is the target function averaged to a high degree. This is the ideal starting point: a function with a unique local minimum. The hollow circle is the starting point, chosen at random. The filled circle is the minimum found using conjugate gradient minimization. (b) Shown is the target function averaged to a lesser extent than that in (a). The hollow point is the start point for the second round of conjugate gradient minimization. This point is the stopping point from (a), i.e., the filled circle in (a). The filled circle is the minimum found using conjugate gradient minimization starting from the hollow circle. (c) Shown is the actual target function. The filled circle is the final stopping point which in this case is the global minimum.

erties that such a function must have to be adequate for this purpose are it must be nonnegative, for each σ it must integrate to 1.0, and as $\sigma \to 0$, h_σ, must converge to a Dirac delta measure, i.e., as $\sigma \to 0$, $S_\sigma \to S$ for any function S. The choice (6) of the function h_σ is merely one of convenience.

To optimize S, a sequence, $\sigma_1 > \sigma_2 > \cdots > \sigma_N$, is chosen, and a local minimum, X_k^*, of $S_{\sigma k}$ is found using conjugate gradient minimization starting from the point X_{k-1}^*. Rather than working directly with (5), the gradient of S_σ is calculated via

$$\nabla S_\sigma(X) = \int S(Y)\nabla_X h_\sigma(X-Y)\,dY = -\int S(Y)\left(\frac{X-Y}{\sigma^2}\right)h_\sigma(X-Y)\,dY. \quad (7)$$

The change of variables $Z = (X - Y)/\sigma$ transforms (7) into

$$\nabla S_\sigma(X) = -\frac{1}{\sigma}\int ZS(X-\sigma Z)h_1(Z)\,dZ. \quad (8)$$

Unfortunately, the integrals in (5) and (8) are too complicated to evaluate analytically for most functions of interest. Thus, instead of an analytical evaluation of (8), a Monte Carlo estimate is used to approximate ∇S_σ. Precisely, an unbiased estimator of ∇S_σ is given by

$$\nabla S_\sigma(X) \cong \xi = -\frac{1}{\sigma M}\sum_{i=1}^{M} Z_i S(X-\sigma Z_i) \quad (9)$$

where $Z_1, \ldots, Z_M$ are N-dimensional independent random variables with the h_1 distribution. This is a point of departure from the DEM algorithm that approximates the required integrals by approximating S with a series of polynomials.

The SAS algorithm generates a sequence of points, $X_1, X_2, \ldots, X_n, \ldots$, in parameter space in the following manner. Given the current point X_n in the search space and the current value of σ, the next point X_{n+1} is determined via

$$X_{n+1} = X_n - \delta P_{n+1} \quad (10)$$

where P_{n+1} is the momentum vector defined by

$$P_{n+1} = (1-\tau)P_n + \tau\xi_n \quad (11)$$

where ξ_n is the gradient estimator at the point X_n [see eq. (9)]. The numbers δ and τ, which are positive and less than 1, are interpreted roughly as step sizes. These parameters are adjusted during each run via statistical tests.[10] The value of σ is decreased after either a fixed number of steps or when P_n becomes sufficiently small.

Unfortunately, because the van der Waals potentials are so steep, the integrals in (5) and (8) do not exist for the scoring function (1) used in this study. To overcome this problem the integrals in (5) are replaced by

$$S_\sigma(X) = -RT\ln\left(\int e^{-S(Y)/RT}h_\sigma(X-Y)\,dY\right) \quad (12)$$

with accompanying changes to (7), (8) and (9). This again is a departure from DEM. Equation (12) should be interpreted as averaging in probability space. For the present work a value of $RT = 0.66$ was used.

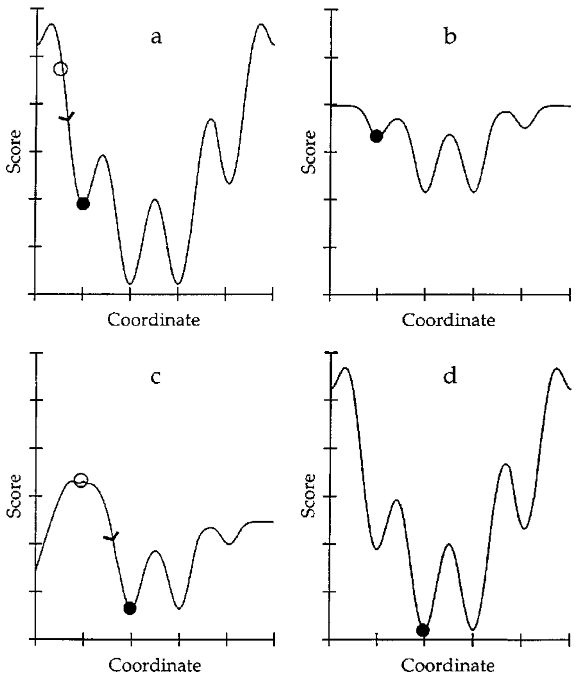

FIGURE 3. A schematic description of the terminal repeller unconstrained subenergy tunneling algorithm. (a) Shown is the target function. The starting point, chosen at random, is the hollow circle. Steepest descent optimization is then applied to find the filled circle. (b) Shown is the transformed function. The transformation has the properties that, first, points that are far above the current local minimum (filled circle) are essentially truncated to 0, and second, that the ordering of the functional values are not changed, i.e., if $f(x) < f(y)$ then $f_T(x) < f_T(y)$, where f is the target function and f_T is the transformed function. (c) Shown is the transformed function along with the penalty term. The penalty term changes the current local minimum into a local maximum. The hollow circle is the starting point and the filled circle the ending point for the second round of steepest descent minimization. (d) Shown is the target function. The filled sphere is the final stopping point found via a final round of steepest descent minimization starting at the stopping point from (c).

The final algorithm considered in this article is the terminal repeller unconstrained subenergy tunneling[12, 13] (TRUST) algorithm. This algorithm has been used in areas such as exploratory seismology. Heuristically, given a starting point in the search space a local minimum is found using steepest descent minimization (Fig. 3a). Then the energy landscape is transformed in such a way that ordering is preserved and any points that lie high above the current local minimum are essentially truncated to zero (Fig. 3b). In addition, a penalty function is added so that the current local minimum becomes a local maximum (Fig. 3c). With this transformed energy function, steepest descent is again applied to find a new local minimum (Fig. 3d).

Mathematically, a sequence of local minima, $X_1^*, X_2^*, \ldots, X_n^*, \ldots$, of S are found as follows. Given the local minimum, X_n^*, the score is transformed via

$$S_T(X) = -\log\left(1 + Ce^{-(S(X)-S(X_n))}\right) - k|X - X_n|^{4/3}\Theta\left(S(X) - S(X_n)\right) \quad (13)$$

where C and k are positive constants and Θ is the heavy side function, i.e., 0 when its argument is negative and 1 when its argument is positive. The first term on the right-hand side of eq. (13) is the transformation of the energy landscape. The second term on the right-hand side of eq. (13) is the penalty term that changes the previously found local minimum into a local maximum. From the current local minimum, X_n^* of S, a local minimum, Y_{n+1}^*, of S_T is found via the steepest decent. Then a local minimum, X_{n+1}^*, of S is found via steepest descent starting from Y_{n+1}^*.

IMPLEMENTATION DETAILS

For each test case the grid dimensions were chosen so that there was at least 5 Å of extra space in all directions around the molecule being docked. This resulted in side lengths that varied from 20 to 30 Å. For each run with each algorithm, the molecule was initially given a uniformly distributed random starting point in the grid box. At the end of each run the position of the molecule was returned to where the best score was found and steepest descent minimization was applied to ensure the molecule was in a local minimum. In all cases 50 docking runs were performed with each algorithm. Because there is no way to guarantee that the global minimum has been found, the best score found from all the runs was assumed to be the global minimum. To ensure that we found the global minimum, we checked the score of the crystallographic coordinates and the score after a gradient minimization starting from the crystallographic coordinates. In all cases, the best scores were found via the docking runs. All docking runs were performed using one CPU of a Dec alpha 4100/400.

The parameters that determine each run were chosen via many tests using galactose in the GM1 binding site of each of the five B subunits of the AB_5 heat labile enterotoxin.[18] We felt this would be a good case to parameterize the algorithms because the binding mode of galactose involves many hydrogen bonds with the hydroxyl groups as well as recognition of the sugar by a tryptophan in the

binding site. In addition, with six rotatable bonds, galactose exhibits a moderate amount of flexibility. The parameters were then used without modification in the test cases described in this work. The reason for choosing the parameters in this manner is that this approach is likely to give a reasonable set of parameters without biasing the choice of the parameters to the actual test cases. The exact parameters used for each algorithm are described below.

A single run for the RW algorithm consisted of 100 cycles each with 10,000 steps. At the end of each cycle the maximum step size was decreased by a factor of 0.99. The initial maximum step size was 10% of the given parameter width. A parameter width is defined simply as the maximum value a parameter can take minus the minimum value the parameter can take. Thus, if the x-coordinate is allowed to vary from -5.0 to 20.0 the parameter width for the x-coordinate is 25, and initially the x-coordinate would be randomly perturbed with a maximum of 2.5. Similarly, a dihedral angle would initially be randomly perturbed with a maximum of 36°, i.e., 10% of 360°.

A single run for the MMCSA algorithm consisted of 100 cycles each consisting of 10,000 steps. At the end of each cycle the maximum step size was reduced by a factor of 0.99, and the temperature was reduced by a factor of 0.95. The initial maximum step size was 10%, the given parameter width. The initial temperature was $RT = 500$ [see eq. (4)].

A single run for the SAS algorithm consisted of 150 cycles each with a maximum of 6000 stochastic conjugate gradient steps [see eqs. (9), (10), and (11)] with only two points being used in the estimate for the gradient, i.e., $M = 2$ in (9). If the momentum vector, described in (11), was below 0.01, the cycle was ended before the maximum number of steps. In virtually all cases each cycle ended well before the maximum number of steps was reached. Initially, the value of σ was taken to be 3.0. At the end of each cycle, σ was reduced by a factor 0.93.

A single run for the TRUST algorithm consisted of 40 cycles, each which located 400 different local minima of S. The value of C in (13) was taken to be 1.0. Each time a new local minimum was found a new value of k in (13) was chosen randomly using a log normal distribution where the underlying normal distribution initially had a mean of 0.7 and a standard deviation of 0.1. This choice of mean and standard deviation allows k to vary roughly between 1.5 and 20. The mean of the normal distribution was decreased by 0.006 at the end of each cycle.

Results

GLOBAL OPTIMIZATION ASPECTS

The first test case was β-trypsin/benzamidine (PDB code 3ptb[19]). Benzamidine (Fig. 4a) is a fairly small (molecular weight of ~100) and completely rigid molecule. Benzamidine binds to β-trypsin with a K_i of 18 μM. The binding mode involves a salt bridge, several hydrogen bonds, and a hydrophobic pocket. Because benzamidine is fairly small and rigid, this is a fairly straightforward test case. In a database of small molecules to be docked, many compounds fall into this category. Thus, it is important to determine how much time is required to reliably dock compounds in this class.

Initially, benzamidine was docked 50 times by each algorithm, allowing 10^6 scoring function evaluations per run. In this case, the RW, MMCSA, and SAS algorithms successfully docked benzamidine to within 5% (~2 kcal/mol) of the global minimum in all 50 runs. In contrast, the TRUST algorithm succeeded in only 37 of the 50 runs. The fact that the RW algorithm succeeded in all 50 runs indicated that 10^6 functional evaluations was too many for this case. With this in mind, the success distributions, i.e., the probability that a given algorithm succeeded as a function of the number of energy evaluations, were examined. As described below, these distributions proved to be extremely informative.

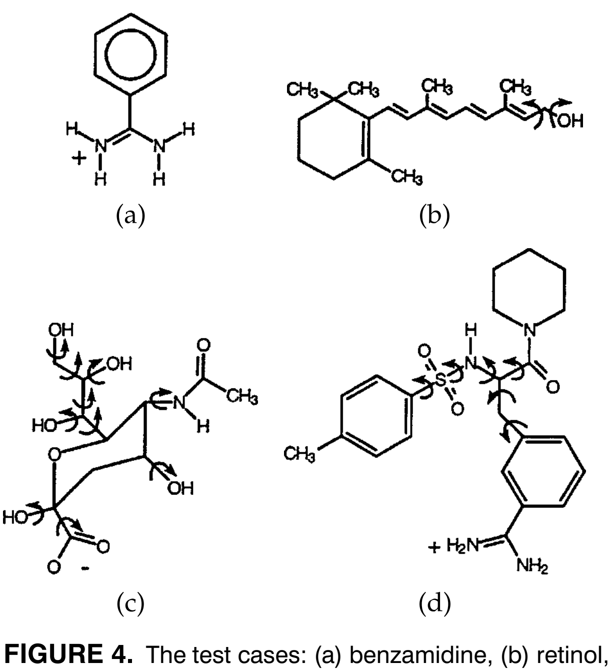

FIGURE 4. The test cases: (a) benzamidine, (b) retinol, (c) sialic acid, (d) 3-TAPAP.

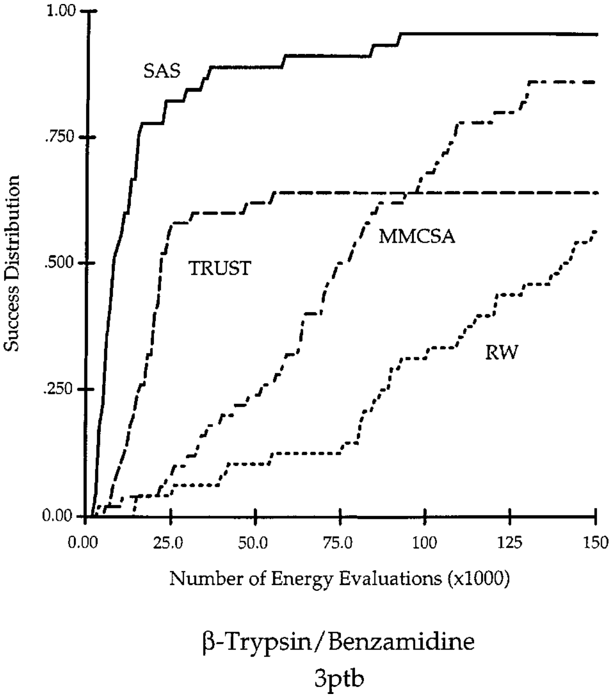

FIGURE 5. The success distributions for docking benzamidine. These curves should be interpreted as the probability that a given algorithm has successfully docked the molecule (y-axis) after the given number of scoring function evaluations (x-axis). For this case, a docking run was considered successful when it found a position whose score was within 5% (~2 kcal/mol) of the global minimum. To create these curves, the best score found was saved every 1000 function evaluations for each run with each algorithm. In this case, the global minimum is approximately 0.54 Å from the position of benzamidine in the crystal structure.

As the success distributions for benzamidine show (Fig. 5), the SAS algorithm was in this case superior to the other algorithms. The SAS algorithm achieved 50% success by 10^4 functional evaluations compared to 2.2×10^4 for TRUST, 7.4×10^4 for MMCSA, and 14×10^4 for RW, with success being judged as getting within 5% of the global minimum (approximately 2 kcal/mol error). Furthermore, SAS achieved 75% reliability in 2.3×10^4 functional evaluations, while the other algorithms did not achieve this reliability until some time later.

There are two ways to achieve higher degrees of reliability: either allow longer runs, or allow multiple independent runs. Clearly, at a certain point the second option becomes more efficient than the first. For example, seven independent runs each with 50% reliability yields a reliability of $[1 - (1 - 0.5)^7]$, i.e., better than 99% reliability. For benzamidine the score can be evaluated 16,000 times per second. Thus, SAS can achieve 99% reliability in 4.4 s compared to 9.6 s for TRUST, 32 s for MMCSA, and 61 s for RW.

The second test case was retinol/retinol binding protein (PDB code 1rbp[20]). Retinol (Fig. 4b), which binds to retinol binding protein with a K_i of 190 nM, is again a fairly small (molecular weight of ~250) and rigid (two rotatable bonds) molecule. This case is, however, significantly more difficult than the benzamidine/trypsin case because the retinol binding site is nearly completely enclosed. For example, in solution, retinol has approximately 570 Å^2 of solvent accessible surface area compared to less than 2 Å^2 of solvent-accessible surface area after binding. Thus, to properly dock retinol, an algorithm must pass retinol through the protein. Although it might not be common phenomena for a ligand to be completely buried upon binding, success in this case would be a strong indication that the algorithm can find tight pockets.

In this case, the RW and MMCSA algorithms failed in all 50 runs. The SAS and TRUST algorithms can, however, successfully dock retinol (Fig. 6), again judging success as getting within 5% of the global minimum (approximately 2 kcal/mol error). The SAS algorithm achieved 50% reliability by 0.8×10^4 functional evaluations compared to 7.5×10^4 for TRUST. For retinol, the score can be evaluated 10,000 times per second. Thus, arguing as above, the SAS algorithm can achieve 99% reliability in 5.6 s compared to 52 s for TRUST.

As a side note, the retinol/retinol binding protein case is an excellent case to demonstrate the practical difference between the linear and the logarithmic interpolation schemes. Because retinol is entirely enclosed in the binding site, the interpolation scheme becomes critical because a poor interpolation scheme could easily hide the binding pocket. The comparison between the two interpolation schemes using the SAS algorithm is shown in Figure 6. Although the difference between the interpolation schemes is small, the difference in the results is significant. With the logarithmic interpolation the SAS algorithm achieved 50% reliability in 0.8×10^4 functional evaluations compared to a 40% reliability after 1.0×10^4 functional evaluations with linear interpolation. The score using the linear interpolation is no faster to evaluate than that with the logarithmic interpolation. Thus, to achieve 99% reliability SAS required only 5.6 s with the logarithmic interpolation compared to 9 s with linear interpolation.

The third test case was sialic acid/hemagglutinin (PDB code 4hmg[21]). Sialic acid (Fig. 4c) has a mole-

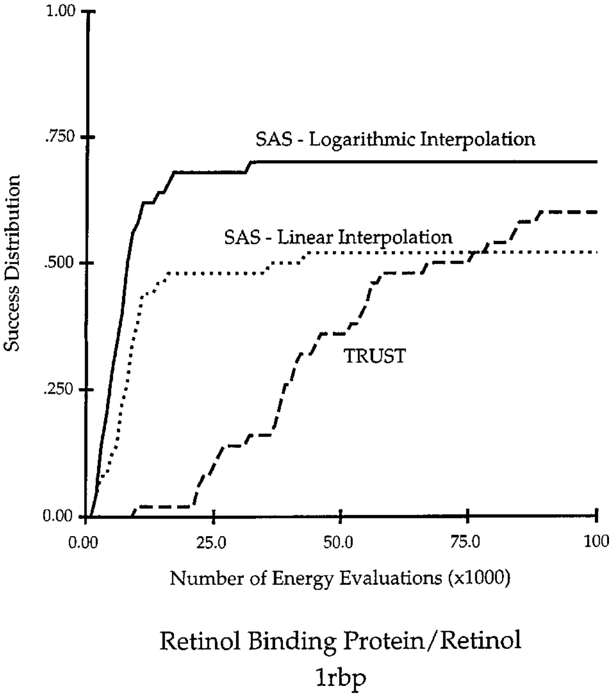

FIGURE 6. The success distributions for docking retinol. Here, any run that came within 5% (~2 kcal/mol) of the global minimum was deemed successful. The success distributions for the RW and MMCSA algorithms are not shown because these algorithms were never able to approach the global minimum. In this case, the global minimum is approximately 0.7 Å from the position of retinol in the crystal structure. These curves were created as those in Figure 5.

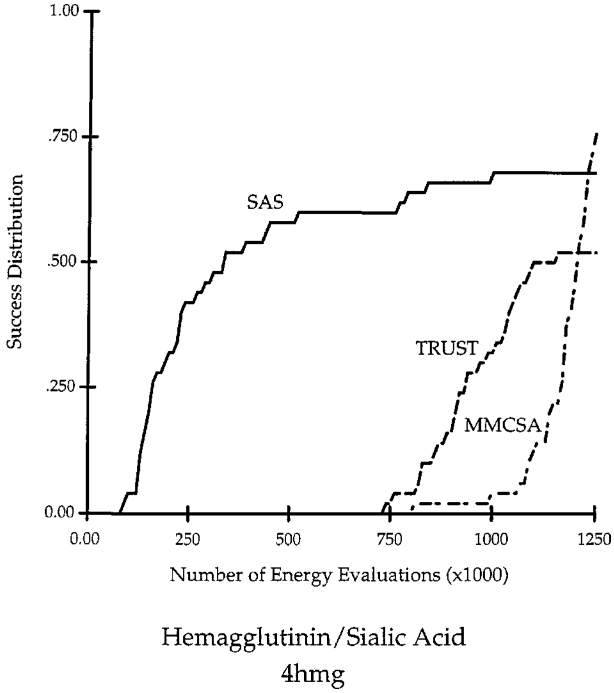

FIGURE 7. The success distributions for docking sialic acid. Here a docking run using the SAS algorithm was considered successful if it came within 5% (~2 kcal/mol) of the global minimum. A docking run with either the MMCSA or TRUST algorithm was considered successful if it came within 20% (~10 kcal/mol) of the global minimum. In this case, the global minimum is approximately 1.25 Å from the crystallographic conformation.

cular weight of ~280 and 10 rotatable bonds, and binds to hemagglutinin through a network of hydrogen bonds with a K_i around 1 mM.[22] As a result, sialic acid provides a significantly more difficult test case. In terms of flexibility, this compound approaches the limits of what is considered a reasonable lead compound in a drug design study.

In this case, the RW algorithm never came close to finding the global minimum. The MMCSA and TRUST algorithms came within 20% of the global minimum (approximately 10 kcal/mol error), with a reliability of 25% after nearly 100×10^4 functional evaluations (Fig. 7). The SAS algorithm, however, was able to come within 5% of the global minimum (approximately 2 kcal/mol error), with a reliability of 50% after 33×10^4 functional evaluations (Fig. 7). The score for sialic acid can be evaluated 3500 times per second. Thus, SAS in this case can achieve 99% reliability in 11 min.

The final test case was 3-TAPAP/trypsin (PDB code 1pph[23]). TAPAP (Fig. 4d) has a molecular weight of ~400 and six rotatable bonds. Trypsin and 3-TAPAP bind with a K_i of 1.2 μM. The binding mode is determined by a combination of hydrogen bonds and hydrophobic interactions, with the tosylate group being packed against the piperidine ring. Although 3-TAPAP has fewer rotatable bonds than sialic acid, a change in any of its dihedral angles will significantly alter its shape, whereas for sialic acid, many of its rotatable bonds are hydroxyls and do not affect to a significant extent its shape. As a result, this case is probably the most difficult of the four, and certainly approaches the limits of what is considered a decent starting compound in a drug design project.

In this case, the RW, MMCSA, and TRUST algorithms rarely came within 10% (approximately 10 kcal/mol error) of the global minimum. The SAS algorithm did, however, come within 5% of the global minimum (approximately 5 kcal/mol error) with 50% reliability after 33×10^4 functional evaluations. Because the score for 3-TAPAP can be

evaluated 3200 times per second, SAS can in this case achieve 99% reliability in 12 min.

THE RELEVANCE AND DISTRIBUTION OF THE FINAL DOCKED CONFORMATIONS

As a final comparison of the algorithms we consider the distributions of the docked conformations in terms of energies, the conformations relative to each other, and the conformations relative to the crystallographic coordinates. These distributions are given in Table I. This table clearly confirms the results of the previous section: the SAS algorithm consistently finds lower energies than the other three algorithms.

Beyond its ability to optimize the energy, an algorithm can be judged by its ability to consistently arrive at a single spacial position. In all four cases, the SAS algorithm consistently found the spacial position corresponding to the global minimum: 100% of the runs with benzamidine, 74% of the runs with retinol, 94% of the runs with sialic acid, and 74% of the runs with 3-TAPAP ended in the top cluster. None of the other three algorithms matched this performance. With the SAS algorithm the cluster width, defined as the average rms deviation of

TABLE I.
Distribution and Relevance of the Final Docked Conformations.[a]

	Benzamidine	Retinol	Sialic Acid	3-TAPAP
Random walk				
Number of clusters	1		26	
Cluster size/rank	50/1		16/1	
Scores (kcal/mol)	−51.5/−50.8		−41.6/−34.9	
RMSD (Å)	0.55/0.34		1.60/1.83	
Cluster width (Å)	0.83/0.33		2.61/1.57	
MMSCA				
Number of clusters	1		6	
Cluster size/rank	50/1		45/1	
Scores (kcal/mol)	−51.8/−50.8		−52.8/−49.1	
RMSD (Å)	0.41/0.40		0.88/0.91	
Cluster width (Å)	0.98/0.44		1.95/0.89	
SAS				
Number of clusters	1	4	3	2
Cluster size/rank	50/1	37/1	47/1	37/1
Scores (kcal/mol)	−52.3/−51.9	−57.7/−57.1	−54.3/52.9	−91.6/−89.8
RMSD (Å)	0.54/0.52	0.74/0.75	1.25/0.94	4.19/4.05
Cluster width (Å)	0.54/0.13	1.67/0.48	1.35/0.79	1.06/0.42
TRUST				
Number of clusters	10	10	21	28
Cluster size/rank	33/1	34/1	26/1	15/2
Scores (kcal/mol)	−52.2/−51.3	−57.5/−56.6	−52.7/−46.8	−84.0/−78.8
RMSD (Å)	0.55/0.47	0.77/0.75	1.07/0.79	1.55/1.69
Cluster width (Å)	0.88/0.50	1.35/0.56	1.39/0.85	2.20/1.18

[a] Final docked conformations were clustered using a cluster radius of 2.0 Å, i.e., conformations within 2.0 Å rms deviation from each other were considered similar (for benzamidine, its twofold symmetry was taken into account). Cluster size is the number of members in the cluster out of a potential maximum of 50 global optimization runs. Scores shows the cluster's best score followed by the mean score of the cluster. RMSD specifies the rms deviation between the crystallographic coordinates and the best scoring member of the cluster followed by the mean of the rms deviations between the crystallographic coordinates and the members of the cluster. Cluster width gives the maximum rms deviation followed by the mean rms deviation between any pair of members of the cluster. In all cases, the cluster chosen was the global minimum cluster except for the case of TRUST/3-TAPAP. In this case, the only cluster with multiple members was the second ranked cluster, which is the one given. The large rms deviation for SAS/3-TAPAP arises mainly from the tosylate group. Without this group the rms deviation is 0.9 Å. Also, in this case the remaining 13 runs ended in a single cluster approximately 5 kcal/mol above the global minimum and approximately 1.7 Å from the crystallographically determined conformation. Finally, the blank areas indicate cases for which the given algorithm produced no clearly favored cluster.

members of the cluster, ranges from 0.08 Å for benzamidine to 0.79 Å for sialic acid, indicating that the top cluster describes a single conformation.

Finally, in three of the cases the global minimum of the energy function corresponds well to the crystallographic conformation: 0.54 Å for benzamidine, 0.74 Å for retinol, and 1.25 Å for sialic acid. For 3-TAPAP, the global minimum is approximately 4.2 Å from the crystallographic conformation. Most of this deviation arises from the tosylate group. Indeed, without this group the rms deviation drops to 0.9 Å. The likely reason this group is difficult to position is that in this case it makes little direct contact with the protein; rather, it packs against the piperidine ring. The score used in this study has little reward for this sort of internal interaction. The SAS algorithm found this conformation in 37 of the 50 runs. The remaining 13 runs ended in a single cluster approximately 5% (5 kcal/mol) below the global minimum and 1.7 Å from the crystallographic conformation.

Conclusions

Even though the difference between logarithmic interpolation and linear interpolation is small, the effects on the docking results of this improved approximation scheme can be significant: in the case of retinol, docking using linear interpolation required nearly twice as much CPU time to gain the same level of reliability as docking using logarithmic interpolation. This result highlights the need for improved approximation methods. In particular, variants of cubic spline interpolation[24] seem promising though the trade off between fewer functional evaluations and the more expensive functional evaluations will have to be examined closely.

The results with the SAS algorithm were encouraging. It appears that with the SAS algorithm a database of 10,000 typical compounds could be reliably docked to a target protein in a few days. There is, however, some work remaining. As is apparent from the four test cases, the number of energy evaluations required to achieve a desired level of reliability depends on the number of rotatable bonds and the size of the molecule being docked. Some work is still needed to determine this dependence.

The results with the TRUST algorithm were disappointing. The reason for the initial interest in this algorithm was because of its superior performance relative to the SAS algorithm in a number of standard test cases.[13] There are several explanations for its poorer performance in these trials, and there are reasons to believe it might still be useful for molecular docking. These are described below.

First, the TRUST algorithm requires the gradient of the scoring function. Technically, the scoring function used in this study does have a gradient defined almost everywhere, but the gradient is not continuous. A smoother scoring function might lead to improved performance of the TRUST algorithm.

Second, for the purposes of this study, the gradient was evaluated numerically. Given the coarseness of the scoring function used, evaluating the gradient numerically probably did not lead to any additional error. The numerical evaluation does, however, require two scoring function evaluations per degree of freedom. Because this scoring function is so inexpensive to evaluate, an analytical evaluation would not have amounted in much savings in CPU time. For more expensive energy functions, the analytical evaluation of the gradient will, however, be much faster than the numerical evaluation. Thus, the TRUST algorithm will not suffer as much in going to more expensive scoring functions.

Future areas of work include considering more global optimization algorithms including among others genetic algorithms and the diffusion equation method.[16, 17] The success with SAS certainly suggests that there are many powerful global optimization algorithms that can be applied to the docking problem, many of which could be better than SAS. Second, the dependence of these results on the scoring function used is a critical question. In particular, will improved approximation schemes improve docking results in general, and can the TRUST algorithm be used more effectively with a smoother scoring function?

References

1. Goodsell, D. S.; Olson, A. J. Proteins 1990, 8, 195.
2. Hart, T. N.; Read, R. J. Proteins 1992, 13, 206.
3. Jones, G.; Willet, P.; Glen, R. C.; Leach, A. R.; Taylor, R. J Mol Biol 1997, 267, 727.
4. Morris, G. M.; Goodsell, D. S.; Halliday, R. S.; Huey, R.; Hart, W. E.; Belew, R. K.; Olson, A. J. J Comp Chem 1998, 18, 1639.
5. Kuntz, I. D.; Blaney, J. M.; Oatley, S. J.; Langridge, R.; Ferrin, T. E. J Mol Biol 1982, 161, 269.
6. Makino, S.; Kuntz, I. D. J Comp Chem 1997, 18, 1812.
7. Miller, M. D.; Kearsley, S. K.; Underwood, D. J.; Sheridan, M. D. J Comp Aided Mol Des 1994, 8, 153.
8. Rarey, M.; Kramer, B.; Lengauer, T.; Klebe, G. J Mol Biol 1996, 261, 470.
9. Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Science 1983, 220, 671.

10. Ruszczynski, A.; Syski, W. IEEE Trans Auto Control 1983, 12, 1097.
11. Styblinski, M. A.; Tang, T. S. Neural Networks 1990, 3, 467.
12. Cetin, B. C.; Barhen, J.; Burdick, J. W. J Opt Theory Appl 1993, 77, 97.
13. Barhen, J.; Protopopescu, V.; Reister, D. Science 1997, 276, 1094.
14. Mehler, E. L.; Solmajer, T. Protein Eng 1991, 4, 903.
15. Rappé, A. K.; Goddard, W. A. J Phys Chem 1991, 95, 3358.
16. Piela, L.; Kostrowicki, J.; Scheraga, H. A. J Phys Chem 1989, 93, 3339.
17. Scheraga, H. A. Biophys Chem 1996, 59, 329.
18. Merritt, E. A.; Sarfaty, S.; Feil, I. K.; Hol, W. G. J. Structure 1997, 11, 1485.
19. Marquart, M.; Walter, J.; Deisenhofer, J.; Bode, W.; Huber, R. Acta Crystallogr B 1983, 39, 480.
20. Jones, T. A.; Newcomer, M. E.; Cowan, S. W. Proteins 1990, 8, 44.
21. Weis, W. I.; Brown, J. H.; Cusak, S.; Paulson, J. C.; Skehel, J. J.; Wiley, D. C. Nature 1988, 333, 426.
22. Watowich, S. J.; Skehel, J. J.; Wiley, D. C. Structure 1994, 2, 719.
23. Turk, D.; Stuerzebecher, J.; Bode, W. FEBS Lett 1991, 287, 133.
24. Oberlin, D.; Scheraga, H. A. J Comp Chem 1998, 19, 71.